Порівняльний аналіз штучних нейронних мереж для вирішення задачі класифікації зображень символів

Єгор Третяков
egorken3v@knu.ua

17 травня 2025 р.

Анотація

У цій роботі проведено порівняльне дослідження трьох сучасних архітектур глибокого навчання — ResNet, MobileNet та Vision Transformer (ViT) — на трьох суттєво різних за структурою та візуальним представленням наборах даних: EMNIST, KMNIST і SVHN. Вибрані датасети охоплюють широкий спектр задач класифікації зображень: від розпізнавання рукописних латинських літер і японських символів до числових позначень у природному візуальному середовищі. Така різноманітність дозволяє комплексно оцінити здатність моделей до узагальнення знань за умов зміни розподілу вхідних даних.

Дослідження зосереджено на аналізі точності класифікації, динаміки навчання та стійкості моделей до зміни домену. Усі архітектури навчено в уніфікованому середовищі з ідентичними умовами попередньої обробки даних та гіперпараметрів оптимізації, що забезпечує об'єктивне порівняння їхньої ефективності. Отримані результати дозволяють сформулювати практичні рекомендації щодо вибору моделі залежно від природи та складності задачі класифікації зображень.

Зміст

1 Вступ

Задача класифікації зображень залишається однією з базових у галузі комп'ютерного зору, а ефективність глибоких моделей значною мірою визначається їх здатністю адаптуватися до особливостей вхідних даних. Хоча сучасні архітектури демонструють високі результати на стандартних еталонних наборах, їх продуктивність на менш однорідних або спеціалізованих даних може суттєво відрізнятися.

Метою цієї роботи є системна оцінка здатності провідних представників згорткових та трансформерних архітектур до адаптації на різнотипних вхідних даних. Зокрема, досліджено три популярні та концептуально різні моделі: ResNet (глибока залишкова згорткова мережа), MobileNet (легка мобільна згорткова архітектура) та Vision Transformer (ViT) — трансформерна модель, адаптована для зображень.

Для проведення повноцінного експериментального порівняння обрано три відкриті набори даних, які суттєво різняться за структурою та складністю: EMNIST (рукописні літери латинського алфавіту), KMNIST (японська канна) та SVHN (цифри у природному контексті). Такий підбір дозволяє проаналізувати, як моделі реагують на зміни візуального представлення, складності вхідних даних та рівня шуму.

У межах дослідження увагу приділено не лише кінцевій точності класифікації, а й динаміці збіжності, здатності до узагальнення в умовах обмежених даних та стійкості моделей до зміни вхідного домену. Це дозволяє отримати глибше розуміння практичної ефективності обраних архітектур у реалістичних умовах.

2 Набори даних

У цьому дослідженні використано три відкриті набори даних, кожен із яких репрезентує окремий підтип задачі класифікації зображень.

2.1 EMNIST (Extended MNIST)

EMNIST — розширений набір даних, що містить рукописні латинські літери та цифри, є доповненням до класичного MNIST. Використовується підмножина *Balanced*, яка охоплює 47 класів. Зображення представлені у відтінках сірого з роздільною здатністю 28×28 пікселів. Приклади зображень з набору даних можна побачити на рис.1. Більш детальна інформація доступна у офіційній документації проєкту [1] та відповідній статті [2].

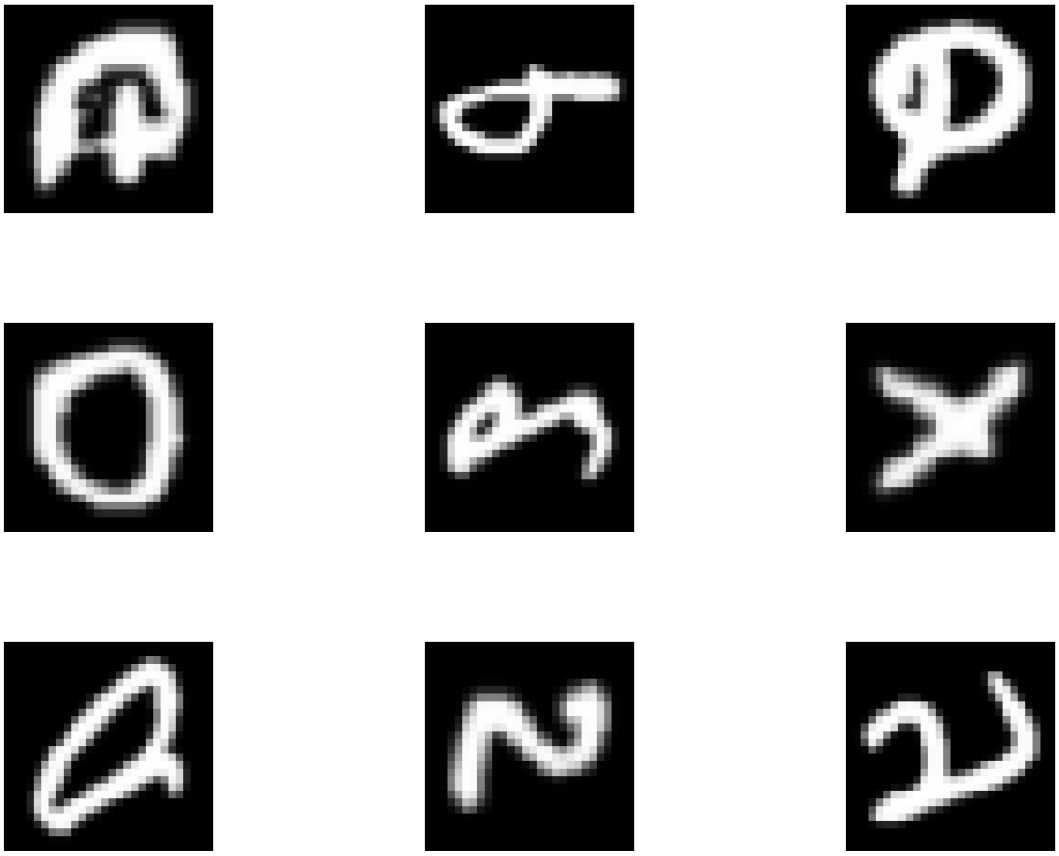

Рис. 1: Приклади зображень із набора даних **EMNIST**.

2.2 KMNIST (Kuzushiji MNIST)

KMNIST — набір рукописних зображень японських символів з кани (hiragana). Містить 10 класів із по 6000 зображень на кожен клас, також із розміром 28×28 пікселів у відтінках сірого. Приклади зображень з набору даних можна побачити на рис.2. Детальніше про датасет можна дізнатися з офіційного репозиторію [3] та відповідній статті [4].

2.3 SVHN (Street View House Numbers)

SVHN — кольоровий набір даних, що складається з зображень цифр, виділених із фотографій будинків, зроблених у реальному середовищі. Містить 10 класів (цифри 0–9) з роздільною здатністю 32×32 пікселі. Приклади зображень з набору даних можна побачити на рис.3. Набір є більш складним через наявність фону, шуму і різної якості зображень. Офіційний опис доступний за посиланням [5].

Різноманітність цих наборів забезпечує комплексну перевірку здатності моделей до узагальнення та адаптації у різних доменах.

Рис. 2: Приклади зображень із набора даних **KMNIST**.

3 Моделі штучних нейронних мереж

Для порівняння обрано три сучасні архітектури глибокого навчання, які репрезентують класичні згорткові нейронні мережі (CNN) і новітні трансформерні підходи до обробки зображень.

3.1 ResNet (Residual Network)

ResNet — це архітектура, розроблена для вирішення проблеми затухання градієнту у дуже глибоких нейронних мережах. Вона вводить концепцію *залишкових зв'язків* (skip connections), які додають вихід певного шару до виходу наступного шару, що дає змогу градієнтам ефективно проходити через багато шарів під час зворотного поширення.

У дослідженні застосовано варіант `ResNet-18`, який має 18 шарів та складається з послідовності залишкових блоків, що складаються із згорткових шарів, нормалізації та функцій активації ReLU. Такий підхід дозволяє моделі мати достатню глибину для виловлювання складних

Рис. 3: Приклади зображень із набора даних **SVHN**.

патернів, при цьому не створюючи надмірної обчислювальної складності.

Детальний опис архітектури наведено у статті [6], а офіційна реалізація доступна у документації PyTorch [7].

3.2 MobileNetV2

MobileNetV2 — архітектура, розроблена спеціально для мобільних пристроїв і вбудованих систем, де обмежені обчислювальні ресурси та енергоспоживання є критичними.

Основою моделі є *глибинні згортки* (depthwise separable convolutions), які розбивають звичайну згортку на дві операції: окремі згортки по кожному каналу і точкові (pointwise) згортки, що значно зменшує кількість параметрів та обчислень.

Ключовою інновацією MobileNetV2 є *інвертовані залишкові блоки* (inverted residual blocks) з лінійною активацією в кінці блоку, що покращує передачу інформації і зменшує втрати корисної інформації у глибо-

ких шарах.

Детальний опис архітектури наведено в оригінальній роботі [8], а код та приклади — у документації PyTorch [7].

3.3 ViT (Vision Transformer)

ViT — це модель, яка вперше застосувала трансформерну архітектуру, успішну в обробці природної мови, до завдань комп'ютерного зору.

Модель розбиває вхідне зображення на фіксовані патчі (наприклад, 16×16 пікселів), кожен з яких перетворюється у вектор-ембеддинг. Ці вектори подаються у трансформер як послідовність токенів, до яких додаються позиційні кодування для збереження просторової інформації.

Самоувага (self-attention) дозволяє моделі враховувати глобальні залежності між різними частинами зображення, що є перевагою у порівнянні зі згортковими мережами, які мають локальний характер обробки.

ViT показав конкурентну точність на великих датасетах і задає новий тренд у комп'ютерному зорі.

Базові принципи описані у статті [9], реалізація доступна через репозиторій HuggingFace [10] та сторонні PyTorch-бібліотеки [11].

Усі архітектури використовувалися з попередньо навченими вагами, адаптованими під кількість класів кожного датасету. Навчання та тестування здійснювалися в уніфікованому середовищі з однаковими гіперпараметрами (оптимізатор, learning rate, кількість епох), що дозволяє об'єктивно порівнювати якість та стабільність моделей на різних типах вхідних даних.

4 Методика тестування

Для об'єктивної оцінки продуктивності моделей у цій роботі застосовано уніфікований підхід до тестування, що забезпечує реплікованість і порівнянність результатів. Код реалізації експериментів, налаштування моделей та скрипти для повторення дослідження доступні на GitHub [12].

4.1 Попередня обробка даних

Вхідні зображення нормалізовано до діапазону $[0, 1]$ шляхом поділу піксельних значень на 255. Для кольорового набору SVHN використовувалась перетворення до трьох каналів RGB, тоді як для EMNIST і KMNIST — одиночний канал градацій сірого.

4.2 Розподіл даних

Для кожного набору даних використано стандартний поділ на тренувальну, валідаційну та тестову множини, якщо такий передбачений. Якщо валідаційної множини немає, частина тренувальних даних (10–15%) виділялася для валідації.

4.3 Параметри навчання

Всі моделі навчалися із застосуванням однакових гіперпараметрів:

- Оптимізатор: Adam із початковою швидкістю навчання 10^{-3},
- Розмір батчу: 64,
- Кількість епох: 15,
- Регуляризація: відсутня (для чистоти порівняння),
- Функція втрат: категоріальна крос-ентропія.

4.4 Оцінка якості моделей

Для вимірювання точності класифікації застосовано метрику **accuracy** — відношення правильно класифікованих прикладів до загальної кількості в тестовому наборі.

Окрім accuracy, для більш глибокого аналізу якості класифікації розраховували такі метрики як **Precision** (точність), **Recall** (повнота) та **F1-score** — гармонійне середнє між точністю і повнотою. Ці показники дозволяють оцінити баланс між правильними позитивними прогнозами та пропущеними випадками, що особливо важливо при роботі з дисбалансованими класами.

Додатково проаналізовано динаміку збіжності на тренувальній та валідаційній множинах для оцінки стійкості та уникнення перенавчання, а також матрицю плутанини. Було проведено оцінку розміру моделі та обсягу використаної **VRAM** (відеопам'яті) під час навчання і тестування для визначення ресурсних вимог кожної архітектури.

4.5 Апаратура та програмне забезпечення

Експерименти виконувалися на робочій станції з GPU NVIDIA RTX 4070 SUPER. Використовувались бібліотеки PyTorch (версія 2.7.0) та стандартні інструменти для обробки даних та візуалізації результатів.

5 Результати експериментів

У результаті проведених експериментів було отримано кількісні показники продуктивності трьох архітектур на тестових наборах даних. Нижче наведені результати для трьох датасетів: SVHN, EMNIST та KMNIST. Для кожного датасету наведено основні метрики класифікації: accuracy, F1-міра, precision і recall (Табл.1).

Табл. 1: Порівняння продуктивності моделей на різних датасетах

Датасет	Модель	Accuracy	F1-міра	Precision	Recall
EMNIST	ResNet	94.71%	94.69%	94.82%	94.71%
	MobileNet	95.03%	95.03%	95.09%	95.03%
	ViT	88.89%	88.87%	89.46%	88.89%
KMNIST	ResNet	97.24%	97.23%	97.26%	97.24%
	MobileNet	97.79%	97.79%	97.82%	97.79%
	ViT	90.58%	90.58%	90.71%	90.58%
SVHN	ResNet	94.29%	94.30%	94.46%	94.29%
	MobileNet	95.79%	95.79%	95.82%	95.79%
	ViT	89.7%	89.6%	89.6%	89.7%

Окрім якісних характеристик, було проаналізовано час навчання та максимальні обсяги використаної VRAM під час навчання (τ) та класифікації на тренувальному наборі даних (Табл.2).

Табл. 2: Ресурсні характеристики моделей

Датасет	Модель	τ (с)	Train VRAM (MB)	Test VRAM (MB)
EMNIST	ResNet	2523	876.80	423.50
	MobileNet	4689	2466.56	310.02
	ViT	18157	4945.70	1600.37
KMNIST	ResNet	1360	1268.81	815.51
	MobileNet	2350	2858.38	601.84
	ViT	8881	5337.65	1992.32
SVHN	ResNet	1693	1281.14	827.86
	MobileNet	2892	2870.64	614.10
	ViT	11196	5354.40	2010.57

6 Порівняльний аналіз

У цьому розділі проведено порівняльний аналіз трьох архітектур нейронних мереж (ResNet, MobileNet та ViT) на основі результатів експериментів, наведених у розділі 5. Аналіз охоплює як якісні характеристики точності класифікації, так і ресурсні вимоги кожної моделі.

6.1 Аналіз точності класифікації

Аналізуючи метрики якості класифікації з таблиці 1, можна зробити такі висновки:

MobileNet демонструє найвищу точність класифікації на всіх трьох датасетах (95.03% на EMNIST, 97.79% на KMNIST та 95.79% на SVHN), випереджаючи ResNet на 0.3–1.5 процентних пункти. Архітектура ViT суттєво поступається згортковим моделям із різницею в точності від 4.6% до 7.2%.

Виникає здивування, що Vision Transformer демонструє суттєво гірші результати порівняно зі згортковими мережами. Та це можна пояснити кількома ключовими факторами:

По-перше, ViT потребує значно більших обсягів тренувальних даних для ефективного навчання. Ця модель спроєктована для роботи з великими наборами даних, такими як ImageNet-21k [13] або JFT-300M [14, 15], де вона має змогу навчитися складних патернів завдяки численним прикладам [16]. У задачах із обмеженою кількістю зразків, як у нашому випадку, ViT часто піддається перенавчанню або не здатна знайти оптимальні представлення.

По-друге, трансформери не мають вбудованих індуктивних обмежень, характерних для згорткових мереж, таких як локальна зв'язність і просторові ієрархії. Ці індуктивні обмеження дозволяють згортковим мережам ефективно виявляти важливі ознаки навіть на невеликих датасетах, тоді як ViT повинна навчатися цих властивостей з нуля, що вимагає більше даних і часу [17, 18].

Отже, незважаючи на потенціал трансформерних архітектур, у задачах із відносно невеликими датасетами згорткові мережі (MobileNet та ResNet) залишаються більш ефективними та ресурсозберігаючими рішеннями.

Для всіх моделей спостерігається висока узгодженість між різними метриками (Accuracy, F1-міра, Precision, Recall), що свідчить про збалансовану класифікацію без суттєвого зміщення на користь окремих класів.

6.2 Аналіз ресурсних вимог

Розглядаючи дані таблиці 2, можна виділити такі ключові моменти:

Архітектура ResNet навчається найшвидше на всіх датасетах — приблизно в 1.7–1.9 раза швидше за MobileNet та в 6.5–7.2 раза швидше за ViT. Повільне навчання ViT є суттєвим недоліком при обмежених обчислювальних ресурсах. Це пов'язано з тим, що механізм self-attention у ViT має високу обчислювальну складність — з квадратичною залежністю від розміру вхідного зображення — і потребує значних ресурсів пам'яті. Така особливість ускладнює застосування ViT на обмежених за потужністю системах і робить оптимізацію моделі під час тренування більш складною [19, 16].

ViT споживає найбільше відеопам'яті під час тренування (4.9–5.4 ГБ), що в 1.8–2 раза більше, ніж MobileNet, і в 4.1–5.6 раза більше, ніж ResNet. Найекономнішим у цьому плані є ResNet. При інференсі зберігається дещо інша тенденція: ViT вимагає найбільше ресурсів, MobileNet — найменше, а ResNet коливається від помірного до середнього використання. ResNet споживає менше пам'яті та навчається швидше, ніж MobileNet, хоча має більше параметрів. Водночас MobileNet під час тренування використовує більше пам'яті і часу, але на інференсі (тестовому проході) є найбільш економною моделлю. Це пояснюється архітектурними особливостями та особливостями реалізації:

MobileNet застосовує depthwise separable convolutions, які значно зменшують кількість параметрів і обчислень у порівнянні зі стандартними згортками. Проте ця техніка призводить до більшої кількості дрібних операцій, що створює додаткові накладні витрати на обробку та менш ефективне використання апаратних прискорювачів під час тренування [20, 21].

ResNet використовує класичні згортки та залишкові зв'язки (residual connections), які допомагають стабілізувати і пришвидшити процес навчання, полегшуючи проходження градієнтів. Крім того, згорткові операції у ResNet добре оптимізовані в апаратних бібліотеках (наприклад, cuDNN), що забезпечує ефективніше використання пам'яті і швидшу обробку [22, 23].

Під час інференсу MobileNet споживає менше ресурсів завдяки меншій кількості параметрів і обчислень, що робить його більш економним у порівнянні з ResNet [20, 21].

Тобто, велика кількість дрібних операцій у MobileNet збільшує накладні витрати під час тренування, тоді як ResNet виграє у швидкості й ефективності навчання через оптимізації архітектури та апаратної підтримки.

7 Висновки

Базуючись на проведеному вище аналізі, можна зробити наступні висновки:

Для розгортання на продуктивних системах з обмеженими ресурсами (особливо на мобільних пристроях) оптимальним є використання MobileNet, яка забезпечує найвищу точність при помірних вимогах до пам'яті під час інференсу.

Для швидкого прототипування або навчання на обмежених обчислювальних ресурсах доцільно використовувати ResNet, яка демонструє найшвидше навчання та добрий баланс між швидкістю та точністю класифікації.

Використання архітектури ViT без додаткових оптимізацій, великого обсягу даних або модифікацій не рекомендується через суттєво вищі вимоги до ресурсів та тривалість навчання. Загалом, для задач класифікації зображень на відносно невеликих датасетах згорткові мережі (MobileNet і ResNet) показують кращий баланс між точністю та обчислювальними витратами, ніж трансформерні моделі.

Список використаних джерел

[1] Gregory Cohen та ін. *EMNIST: Extending MNIST to handwritten letters*. https://www.nist.gov/itl/iad/image-group/emnist-dataset. 2017.

[2] Gregory Cohen та ін. «EMNIST: an extension of MNIST to handwritten letters». В: *arXiv preprint arXiv:1702.05373* (2017). URL: http://arxiv.org/abs/1702.05373.

[3] Tanakorn Clanuwat та ін. *KMNIST: Kuzushiji-MNIST, a handwritten Japanese character dataset*. https://github.com/rois-codh/kmnist. 2018.

[4] Thanard Clanuwat та ін. «Deep learning for classical Japanese literature». В: *arXiv preprint arXiv:1812.01718* (2018). URL: https://arxiv.org/abs/1812.01718.

[5] Yuval Netzer та ін. *SVHN: Street View House Numbers Dataset*. http://ufldl.stanford.edu/housenumbers/. 2011.

[6] Kaiming He та ін. «Deep Residual Learning for Image Recognition». В: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), с. 770—778. DOI: 10.1109/CVPR.2016.90.

[7] PyTorch Contributors. *Torchvision Models Documentation*. https://pytorch.org/vision/stable/models.html. Accessed: 2025-05-16. 2025.

[8] Mark Sandler та ін. «MobileNetV2: Inverted Residuals and Linear Bottlenecks». В: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), с. 4510—4520. DOI: 10.1109/CVPR.2018.00474.

[9] Alexey Dosovitskiy та ін. «An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale». В: *International Conference on Learning Representations (ICLR)* (2021). URL: https://openreview.net/forum?id=YicbFdNTTy.

[10] Hugging Face. *Vision Transformer — Transformers Documentation*. https://huggingface.co/docs/transformers/model_doc/vit. Accessed: 2025-05-16. 2025.

[11] Phil Wang. *vit-pytorch: Vision Transformer implemented in Pytorch*. https://github.com/lucidrains/vit-pytorch. Accessed: 2025-05-16. 2025.

[12] Yehor (Tr3tiakoFF) Tretiakov. *NeuroVisionLab: Репозиторій з кодом для досліджень у сфері штучних нейронних мереж*. https://github.com/Tr3tiakoFF/NeuroVisionLab. Доступ 16 травня 2025. 2025.

[13] *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)*. http://image-net.org/challenges/LSVRC/. Accessed: 2025-05-16.

[14] Google AI. *Scaling Vision Transformers*. https://ai.googleblog.com/2021/05/scaling-vision-transformers.html. Accessed: 2025-05-16.

[15] Dhruv Mahajan та ін. «Exploring the Limits of Weakly Supervised Pretraining». В: *arXiv preprint arXiv:1805.00932* (2018). URL: https://arxiv.org/abs/1805.00932.

[16] Alexey Dosovitskiy та ін. «An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale». В: *arXiv preprint arXiv:2010.11929* (2020). URL: https://arxiv.org/abs/2010.11929.

[17] Thomas De Vries та ін. «ConViT: Improving Vision Transformers with Soft Convolutional Inductive Biases». В: *arXiv preprint arXiv:2103.10697* (2021). URL: https://arxiv.org/abs/2103.10697.

[18] Zihang Chen та ін. «Bridging the Gap Between Vision Transformers and Convolutional Neural Networks on Small Datasets». В: *arXiv preprint arXiv:2210.05958* (2022). URL: https://arxiv.org/abs/2210.05958.

[19] Ashish Vaswani та ін. «Attention is All You Need». В: *Advances in Neural Information Processing Systems* 30 (2017). URL: https://arxiv.org/abs/1706.03762.

[20] Andrew G. Howard та ін. «MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications». В: *arXiv preprint arXiv:1704.04861* (2017). URL: https://arxiv.org/abs/1704.04861.

[21] Mark Sandler та ін. «MobileNetV2: Inverted Residuals and Linear Bottlenecks». В: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, с. 4510—4520. URL: https://doi.org/10.1109/CVPR.2018.00474.

[22] Kaiming He та ін. «Deep residual learning for image recognition». В: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, с. 770—778. URL: https://doi.org/10.1109/CVPR.2016.90.

[23] Christian Szegedy та ін. «Inception-v4, inception-resnet and the impact of residual connections on learning». В: *Thirty-first AAAI conference on artificial intelligence*. 2017. URL: https://arxiv.org/abs/1602.07261.